Theoretical Computer Science 410 (2009) 736–744

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Short fail-stop signature scheme based on factorization and discrete logarithm assumptions

Willy Susilo *

Centre for Computer and Information Security Research, ICT Research Institute, University of Wollongong, Australia
School of Computer Science and Software Engineering, University of Wollongong, Australia

ARTICLE INFO

Article history:
Received 19 December 2007
Received in revised form 5 October 2008
Accepted 9 October 2008
Communicated by X. Deng

Keywords:
Fail-stop signatures
Short
Authentication
Proof of forgery
Unconditional security
Computational security

ABSTRACT

Fail-stop signature (FSS) schemes protect a signer against a forger with unlimited computational power by enabling the signer to provide a proof of forgery, if it occurs. A decade after its invention, there have been several FSS schemes proposed in the literature. Nonetheless, the notion of *short* FSS scheme has not been addressed yet. Furthermore, the short size in signature schemes has been done mainly with the use of pairings. In this paper, we propose a construction of *short FSS scheme* based on factorization and discrete logarithm assumption. However, in contrast to the known notion in the literature, our signature scheme does *not* incorporate any pairing operations. Nonetheless, our scheme is the *shortest* FSS scheme compared to all existing schemes in the literature that are based on the same assumption. The efficiency of our scheme is comparable to the best known FSS scheme, that is based on the discrete logarithm assumption.

1. Introduction

Ordinary digital signatures, introduced in the seminal paper of Diffie and Hellman [12], allow a signer with a secret key to sign messages such that anyone with access to the corresponding public key be able to verify authenticity of the message. Security of an *ordinary digital signature scheme* relies on a computational assumption, that is assuming that there is *no* efficient algorithm to solve the hard problem that underlies the security of the scheme. This means that if an enemy can solve the underlying problem, he can successfully forge a signature and there is no way for the signer to prove that a forgery has occurred. To provide protection against such an enemy, fail-stop signature (FSS) schemes have been proposed [17,31]. Loosely speaking, an FSS is a signature scheme augmented such that the signer can prove that a forged signature was not generated by him/her. To achieve this property, the signature scheme is constructed such that there are many secret keys that correspond to the same public key and the sender knows only one of the keys. An unbounded enemy can find all the secret keys but cannot determine which secret key is actually used by the sender. In the case of a forgery, that is signing a message with a randomly chosen secret key, the sender can use his secret key to generate a second signature for the same message. This signature will be different with overwhelming probability from the forged one. The two signatures on the same message can be used as a proof that the underlying computational assumption is broken and the system must be stopped — hence the name *fail-stop*. FSS schemes provide unconditional security for the signer, however security for the receiver is computational and relies on the difficulty of the underlying hard problem. FSS schemes in their basic form are one-time primitives and so the key can be used for signing a single message.

FSS schemes and their variants have been studied by numerous authors (see, for example, [21–23,26–30]). The schemes can be broadly divided into two categories: those based on the hardness of discrete logarithm problem and those based on

* Corresponding address: Centre for Computer and Information Security Research, ICT Research Institute, University of Wollongong, Australia. Tel.: +61 242215535.
E-mail address: wsusilo@uow.edu.au.

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.10.025

the difficulty of factorization. The first scheme that uses factorization as its underlying hard problem was proposed in [17, 30]. However, the signing algorithm in this scheme is very inefficient. In [26,28], RSA-based FSS schemes were proposed. These schemes are attractive because of the way the proof of forgery works, i.e. by revealing the non-trivial factor of the modulus. Nonetheless, their signature size is quite long due to the underlying problem used.

1.1. Short signatures vs. pairings

In recent years, pairings have found various applications in cryptography and have allowed us to construct some new cryptographic schemes (for example, [2–6]). To date, there exist three short signature schemes in the literature. In Asiacrypt 2001, Boneh, Lynn, and Shacham [6] proposed a short signature scheme (BLS scheme) using pairings on certain elliptic and hyperelliptic curves. BLS short signature needs a special hash function, which is known as the `MapToPoint` operation [4]. This hash function is probabilistic and generally inefficient. Independently, in PKC 2004 and Eurocrypt 2004, Zhang, Safavi-Naini and Susilo (ZSS) [33] and Boneh and Boyen (BB) [2] proposed a short signature scheme that does not require the `MapToPoint` operation. The security of the ZSS signature relies on the k-CAA problem, whilst the BB signature relies on the q-SDH problem. Furthermore, Boneh and Boyen proposed a variant of this signature scheme that does not require the random oracle model, and the security of their scheme remains relying on the q-SDH problem.

Throughout the year, there are many short signature schemes with special properties proposed in the literature (eg. [3, 11,13,14,34]). Nevertheless, they use pairings as the essential tool of their construction. The short length obtained on these signature schemes are essentially due to the implementation of the elliptic curve used which enables them to gain a short signature (or *shorter* than the existing schemes in the literature).

Unlike the construction of regular signature schemes, unfortunately FSS schemes cannot really take any advantage from the pairings. This is due to the fact that the enemy of FSS scheme is equipped with an unlimited computational power and hence, the underlying problems proposed in pairings can be easily solved (such as Computational Diffie-Hellman assumption, Bilinear Diffie-Hellman assumption, etc.). Therefore, the size of the secret key involved will remain the same as the size of the secret key in the existing constructions based on either factorization and/or discrete logarithm problem. Hence, constructions based on pairings will merely just moving from the existing factorization and/or discrete logarithm problem to pairing group, which is not very interesting.

1.2. Our contributions

In this paper, we propose a short FSS scheme. Interestingly, our scheme does *not* incorporate any pairing operations (in contrast to any other short signature schemes with/without special property that exist in the literature[1]). We evaluate the efficiency of the scheme and show that it is as efficient as the most efficient discrete logarithm based FSS scheme due to van Heijst and Pedersen [29]. We note that van Heijst and Pedersen's scheme is the existing most efficient scheme based on discrete logarithm assumption, while our construction is the most efficient FSS scheme based on factorization. In practice, the length of the signature produced by our scheme is 302 bits, for the appropriately chosen security parameters.

1.3. Paper organization

The paper is organized as follows. In Section 2, we present the basic concepts and definitions of FSS, and briefly review the general construction and its relevant security properties. In Section 3, we present our short FSS construction based on the factorization and discrete logarithm assumptions, and show that it is an instance of the general construction [17] and hence has provable security. Interestingly, although our scheme relies on the factorization assumption, we use a special type of modulus that allow us to achieve a short signature length. Our special type of modulus is used to build a group of composite order with a prime-order subgroup. This type of modulus has been introduced by Brickell and McCurley [7,8] to construct their identification scheme. We also provide some comparison between our scheme and the existing schemes. For the other factorization based FSS scheme, we only select one of them [28] which represents the size of the signature scheme that uses RSA as its underlying hard problem. We also explain the reason why this scheme is chosen instead of the other schemes, such as [24,26,27]. Finally, Section 4 concludes the paper.

2. Preliminaries

In this section, we briefly recall relevant notions, definitions and requirements of fail-stop signatures and refer the reader to [17,19,20] for a more complete account.

2.1. Notations

The length of a number n is the length of its binary representation and is denoted by $|n|_2$. $p|q$ means p divides q.

[1] We note that there is no existing FSS scheme based on pairings that is more efficient or secure than the existing schemes in the literature yet. Furthermore, several attempts have been made (eg. [9]) to further construct FSS based on pairings but unfortunately it was later shown to be insecure [32]. Moving an existing scheme to the elliptic curve group is certainly possible but the result is not very interesting.

The ring of integers modulo a number n is denoted by Z_n, and its multiplicative group, which contains only the integers relatively prime to n, by Z_n^*. Let N denote the natural numbers.

2.2. Review of fail-stop signatures schemes

Similar to an ordinary digital signature scheme, a fail-stop signature scheme consists of a polynomial time protocol and two polynomial time algorithms.

(1) *Key generation*: is a two party protocol between the signer and the center to generate a pair of *secret key*, s_k, and *public key*, p_k. This is different from ordinary signature schemes where key generation is performed by the signer individually and without the involvement of the receiver.
(2) *Sign*: is the algorithm used for signature generation. For a message m and using the secret key s_k, the signature is given by $y = \text{sign}(s_k, m)$.
(3) *Test*: is the algorithm for testing acceptability of a signature. For a message m, a signature y and a given public key p_k, the algorithm produces a *true* response if the signature is acceptable under p_k. That is $test(p_k, m, y) \stackrel{?}{=} true$.

An FSS also includes two more polynomial time algorithms:

4. *Proof*: is an algorithm for proving a forgery;
5. *Proof-test*: is an algorithm for verifying that the proof of forgery is valid.

A secure fail-stop signature scheme must satisfy the following properties [17,30,19].

(1) If the signer signs a message, the recipient must be able to verify the signature (*correctness*).
(2) A polynomially bounded forger cannot create forged signatures that successfully pass the verification test (*recipient's security*).
(3) When a forger with an unlimited computational power succeeds in forging a signature that passes the verification test, the presumed signer can construct a proof of forgery and convinces a third party that a forgery has occurred (*signer's security*).
(4) A polynomially bounded signer cannot create a signature that he can later prove to be a forgery (*non-repudiability*).

To achieve the above properties, for each public key, there exists many matching secret keys such that different secret keys create different signatures on the same message. The real signer knows only one of the secret keys, and can construct one of the many possible signatures. An enemy with unlimited computing power, although can generate all the signatures but cannot determine which one is generated by the true signer. Thus, it would be possible for the signer to provide a proof of forgery by generating a second signature on the message with a forged signature, and use the two signatures to show the underlying computational assumption of the system is broken, hence proving the forgery.

Security of an FSS can be broken if (1) a signer can construct a signature that he can later prove to be a forgery, or (2) an unbounded forger succeeds in constructing a signature that the signer cannot prove that it is forged. These two types of forgeries are completely independent and so two different security parameters, k and σ, are used to show the level of security against the two types of attacks. More specifically, k is the security level of the recipient and σ is that of the signer. It is proved [17] that a secure FSS is secure against adaptive chosen message attack and for all $c > 0$ and large enough k, success probability of a polynomially bounded forger is bounded by k^{-c}. For an FSS with security level σ for the signer, the success probability of an unbounded forger is limited by $2^{-\sigma}$.

In the following we briefly recall the general construction given in [17] and outline its security properties.

2.3. The general construction

The construction is for a single-message fail-stop signature and uses *bundling homomorphisms*. Bundling homomorphisms can be seen as a special kind of hash functions.

Definition 2.1. [17] A bundling homomorphism h is a homomorphism $h : G \to H$ between two Abelian groups $(G, +, 0)$ and $(H, \times, 1)$ that satisfies the following.

(1) Every image $h(x)$ has at least 2^τ preimages. 2^τ is called *bundling degree* of the homomorphism.
(2) It is infeasible to find collisions, i.e., two different elements that are mapped to the same value by h.

To give a more precise definition, we need to consider two families of groups, $\mathcal{G} = (G_K, +, 0)$ and $\mathcal{H} = (H_K, \times, 1)$, and a family of polynomial-time functions indexed by a key, K. The key is obtained by applying a key generation algorithm $g(k, \tau)$, on two input parameters k and τ. The two parameters determine the difficulty of finding collision and the bundling degrees of the homomorphism, respectively. Given a pair of input parameters, $k, \tau \in N$, firstly, using the key generation algorithm, a key K is calculated and then, G_K, H_K and h_K are determined. For a formal definition of bundling homomorphisms see Definition 4.1 [17].

A bundling homomorphism can be used to construct an FSS scheme as follows.

Let the security parameters of the FSS be given as k and σ. The bundling degree of the homomorphism, τ, will be obtained as a function of σ as shown below.

(1) *Prekey generation*: The center computes $K = g(k, \tau)$ and so determines a homomorphism h_K, and two groups G_K and H_K. Let $G = G_K, H = K_K$ and $h = h_K$.

(2) *Prekey verification*: The signer must be assured that K is a possible output of the algorithm $g(k, \tau)$. This can be through providing a zero-knowledge proof by the center or by testing the key by the signer. In any case the chance of accepting a *bad* key must be at most $2^{-\sigma}$.
(3) *Main key generation* gen_A: the signer generates her secret key $sk := (sk_1, sk_2)$ by choosing sk_1 and sk_2 randomly in G and computes $pk := (pk_1, pk_2)$ where $pk_i := h(sk_i)$ for $i = 1, 2$.
(4) The message space M is a subset of Z.
(5) *Signing*: The signature on a message $m \in M$ is,

$$s = \text{sign}(sk, m) = sk_1 + m \times sk_2$$

where multiplying by m is m times addition in G.
(6) *Testing the signature*: can be performed by checking,

$$pk_1 \times pk_2^m \stackrel{?}{=} h(s).$$

(7) *Proof of forgery*: Given an acceptable signature $s' \in G$ on m such that $s' \neq \text{sign}(sk, m)$, the signer computes $s := \text{sign}(sk, m)$ and $proof := (s, s')$.
(8) *Verifying proof of forgery*: Given a pair $(x, x') \in G \times G$, verify that $x \neq x'$ and $h(x) = h(x')$.

Theorem 4.1 [17] proves that for any family of bundling homomorphisms and any choice of parameters the general construction:

(1) produces correct signature;
(2) a polynomially bounded signer cannot construct a valid signature and a proof of forgery;
(3) if an acceptable signature $s^* \neq \text{sign}(sk, m^*)$ is found the signer can construct a proof of forgery.

Moreover for two chosen parameters k and σ, a good prekey K and two messages $m, m^* \in M$, with $m \neq m^*$, let

$$T := \{d \in G | h(d) = 1 \wedge (m^* - m)d = 0\}. \tag{1}$$

Theorem 4.2 [17] shows that given $s = \text{sign}(sk, m)$ and a forged signature $s^* \in G$ such that $test(pk, m^*, s^*) = ok$, the probability that $s^* = \text{sign}(sk, m^*)$ is at most $|T|/2^\tau$ and so the best chance of success for an unrestricted forger to construct an undetectable forgery is bounded by $|T|/2^\tau$. Thus to provide the required level of security σ, we must choose $|T|/2^\tau \leq 2^{-\sigma}$.

Provable security

This general construction is the basis of all known *provably secure constructions* of FSS. It provides a powerful framework by which proving security of a scheme is reduced to specifying the underlying homomorphism, and determining the bundling degree and the set T. Hence, to prove security of a scheme two steps are required.

(1) showing that the scheme is in fact an instance of the general construction;
(2) determine bundling parameter and the size of the set T.

We note that the second generic construction of FSS can be found in [22]. Nonetheless, we do not employ this generic construction in this paper rather than following the one from [17] since the construction in [22] mainly concentrates on how to achieve a provably secure FSS scheme from an authentication code.

3. A short FSS scheme

In this section we propose a new FSS scheme based on factorization and show that it is an instance of the general construction. Proof of forgery is by revealing the secret key kept by the dealer and so verifying the proof is very efficient.

For simplicity, we describe our scheme with a single recipient model. As in [29], the scheme can be extended to multiple recipient by employing a coin-flipping protocol.

As the other FSS schemes, the basic scheme is *one-time* and can be only used once, however, it is possible to extend the scheme to sign multiple messages [1,10,18,29].

Before describing our scheme, we recall some basic preliminaries and notations as follows [25].

Definition 3.1. [25] Let

$$n = p_1^{e_1} \cdots p_r^{e_r}$$

where p_i denotes a prime number. An Euler number $\phi(n)$ is computed as

$$\phi(n) = p_1^{e_1 - 1}(p_1 - 1) \cdots p_r^{e_r - 1}(p_r - 1).$$

Let $\lambda(n)$ denote the exponent of Z_n^*. Then, by the Chinese remainder theorem

$$\lambda(n) = lcm(\lambda(p_1^{e_1}), \cdots, \lambda(p_r^{e_r}))$$

where for any prime power p^e, we have

$$\lambda(p^e) = \begin{cases} p^{e-1}(p-1) & \text{if } p \neq 2 \text{ or } e \leq 2 \\ 2^{e-2} & \text{if } p = 2 \text{ and } e \geq 3 \end{cases}$$

and

$$lcm(a, b) = \frac{ab}{gcd(a, b)}.$$

Applying Definition 3.1 to the case where $n = pq$, where p and q are prime numbers, we obtain the following definition.

Definition 3.2. Let $n = pq$, where p, q are prime numbers which are not 2. Then, $\phi(n) = (p-1)(q-1)$. Furthermore, we have

$$\begin{aligned}\lambda(n) &= lcm(\lambda(p), \lambda(q)) \\ &= lcm((p-1), (q-1)) \\ &= \frac{(p-1)(q-1)}{gcd((p-1), (q-1))} \\ &= \frac{\phi(n)}{gcd(p-1, q-1)}.\end{aligned}$$

Now we are ready to describe our scheme as follows.

Model

There is only a single recipient, $\mathcal{R}$ who also plays the role of the trusted center and performs prekey generation of the scheme.

Prekey generation

Given the two security parameters k and σ, $\mathcal{R}$ chooses two large primes p and q, where $p = c_1\beta p' + 1$, $q = c_2\beta q' + 1$, p', q', β are also prime, $(c_1, c_2) \in Z$ and $gcd(c_1, c_2) = 2$ (which means that both $c_1, c_2 = 2\tilde{c}, \tilde{c} \in Z$). For simplicity, assume $c_1 = 2$ and $c_2 = 4$. To guarantee security, $|\beta|_2$ must be chosen such that the subgroup discrete logarithm problem for the multiplicative subgroup of order β in Z_n^* be intractable (for example, $|n|_2 \approx 1881$ bits and $|\beta|_2 \approx 151$ bits [15]). $\mathcal{R}$ computes $n = pq$, and selects an element α such that the multiplicative order of α modulo n is β, and $gcd(\alpha, c_1c_2\beta^2p'q') = 1$. Note that $\phi(n) = c_1c_2\beta^2p'q'$, and $\lambda(n) = \frac{c_1c_2\beta^2p'q'}{\beta}$. Since $gcd(c_1, c_2) = 2$, we can let $c_1 = 2c_1'$ and $c_2 = 2c_2'$ and further obtain $\lambda(n) = \frac{4c_1'c_2'\beta^2p'q'}{2\beta} = 2c_1'c_2'\beta p'q'$ (See Justification 3.1 for the detail). Let N_β denote the subgroup of Z_n^* generated by α. $\mathcal{R}$ also chooses a secret random number $a \in N_\beta$ and computes $\gamma = \alpha^a \pmod n$. $(\alpha, \beta, \gamma, n)$ is published and (p, q, a) is kept secret. We note that although the factors of n are of a particular form, to our knowledge there is *no* known efficient algorithm for factorization that can be applied in this case. We note that similar construction is used in [7,8]. We also note that when $c_1 = c_2 = 2$, the construction of n coincides with [7,8].

Justification 3.1. *For the above setting, we obtain* $\lambda(n) = 2c_1'c_2'\beta p'q'$.

Proof. Let $p = c_1\beta p' + 1, q = c_2\beta q' + 1, p', q', \beta$ are also prime, and $gcd(c_1, c_2) = 2$. Applying Definition 3.2, we obtain the following.

$$\phi(n) = (p-1)(q-1) = c_1\beta p'c_2\beta q' = c_1c_2\beta^2p'q'.$$

Now, since $gcd(c_1, c_2) = 2$, we can let $c_1 = 2c_1'$ and $c_2 = 2c_2'$. Note that $gcd(c_1, c_2) = gcd(2c_1', 2c_2')$ will remain as 2. If we re-compute $\phi(n)$ using this setting, we will obtain

$$\phi(n) = c_1c_2\beta^2p'q' = 2c_1'2c_2'\beta^2p'q' = 4c_1'c_2'\beta^2p'q'.$$

Furthermore,

$$gcd(p-1, q-1) = gcd(c_1\beta p', c_2\beta q') = gcd(2c_1'\beta p', 2c_2'\beta q') = 2\beta.$$

Hence, we can compute $\lambda(n)$ using Definition 3.2 as

$$\lambda(n) = \frac{\phi(n)}{gcd(p-1, q-1)} = \frac{4c_1'c_2'\beta^2p'q'}{2\beta} = 2c_1'c_2'\beta p'q'$$

as claimed. ■

Lemma 3.1. *It is easy for* $\mathcal{R}$ *to find an element* α *where* $ord_n(\alpha) = \beta$, *for* $p = c_1\beta p' + 1$ *and* $q = c_2\beta q' + 1$ *and* $gcd(c_1, c_2) = 2$, *when* $\mathcal{R}$ *knows the factorization of n.*

Proof. To find an element α where $ord_n(\alpha) = \beta$, $\mathcal{R}$ will perform the following.

(1) Compute $\lambda(n) = 2c_1'c_2'\beta p'q'$, where $c_1' = \frac{c_1}{2}$ and $c_2' = \frac{c_2}{2}$.
(2) Find an element $g \in Z_n^*$ of order $\lambda(n)$. To do so, $\mathcal{R}$ can randomly choose an element $g \in Z_n^*$, find its order and if not equal to $\lambda(n)$, choose another value. The algorithm is efficient because $ord_n(g)|\phi(n)$ and $\phi(n)$ has small number of factors.

(3) Set

$$\alpha = g^{2c'_1c'_2p'q'} \pmod{n}.$$

It is easy to see that $ord_n(\alpha) = \beta$. ■

Prekey verification

Prekey verification will be done by the signer $\mathcal{S}$ by verifying

$$\alpha^{\beta} \stackrel{?}{=} 1 \pmod{n} \quad \text{and} \quad \alpha \neq 1 \pmod{n}.$$

A prekey is *good* if the above equation holds.

Key generation

$\mathcal{S}$ selects $a_1, a_2, b_1, b_2 \in Z_\beta$ as his secret key and computes

$$\eta_1 = \alpha^{a_1}\gamma^{a_2} \pmod{n} \quad \text{and} \quad \eta_2 = \alpha^{b_1}\gamma^{b_2} \pmod{n}.$$

The public key is (η_1, η_2).

Signing a message m

To sign a message $m \in Z_\beta$, $\mathcal{S}$ computes

$$s_1 = a_1 + b_1 m \pmod{\beta} \quad \text{and} \quad s_2 = a_2 + b_2 m \pmod{\beta}$$

and publishes (s_1, s_2) as his signature on m.

Verifying a signature

A signature (s_1, s_2) on a message m passes the verification test if

$$\eta_1\eta_2^m \stackrel{?}{=} \alpha^{s_1}\gamma^{s_2} \pmod{n}$$

holds.

The verification algorithm works because

$$\begin{aligned}\eta_1\eta_2^m \pmod{n} &= \alpha^{a_1}\gamma^{a_2}\left\{\alpha^{b_1}\gamma^{b_2}\right\}^m \pmod{n}\\ &= \alpha^{a_1+b_1m}\gamma^{a_2+b_2m} \pmod{n}\\ &= \alpha^{s_1}\gamma^{s_2} \pmod{n}. \quad ■\end{aligned}$$

Proof of forgery

If there is a forged signature (s'_1, s'_2) which passes the verification test, then the presumed signer can generate his own signature, namely (s_1, s_2), on the same message, and the following equation will hold:

$$\begin{aligned}&\alpha^{s_1}\gamma^{s_2} = \alpha^{s'_1}\gamma^{s'_2} \pmod{n}\\ &\alpha^{s_1+as_2} = \alpha^{s'_1+as'_2} \pmod{n}\\ &\alpha^{s_1-s'_1} = \alpha^{a(s'_2-s_2)} \pmod{n}\\ &s_1 - s'_1 = a(s'_2 - s_2) \pmod{\beta}\\ &a = (s_1 - s'_1)(s'_2 - s_2)^{-1} \pmod{\beta}.\end{aligned}$$

By evaluating a, $\mathcal{S}$ can show that he can solve an instance of discrete logarithm problem which was assumed to be hard.

Proof. From the above proof of forgery steps, it is true that

$$\begin{aligned}&\alpha^{s_1-s'_1} = \alpha^{a(s'_2-s_2)} \pmod{n}\\ &s_1 - s'_1 = a(s'_2 - s_2) \pmod{\beta}\end{aligned}$$

because $ord_n(\alpha) = \beta$. ■

3.1. Security proof

Firstly, we show that the scheme is an instance of the general construction proposed in [17] with the following underlying bundling homomorphism family.

Bundling homomorphism

- *Key Generation:* On input the security parameters k and σ, two primes p and q with $|q|_2 = \sigma$ and $|p|_2 \approx |q|_2$, $p = c_1\beta p' + 1$; $q = c_2\beta q' + 1$; $gcd(c_1, c_2) = 2$; $(c_1, c_2) \in Z$; and an element α where $ord_n(\alpha) = \beta$ are chosen. Let $\gamma = \alpha^a \pmod n$. The key will be $(p, q, \alpha, \beta, \gamma)$.
- *Families of Groups:* Let $n = pq$. Define $G_K = Z_\beta$ and $H_K = Z_n^*$. The homomorphism $h_{(p,q,\alpha,\beta,\gamma)}$ is

$$h_{(p,q,\alpha,\beta,\gamma)} : Z_\beta \times Z_\beta \to Z_n^*,\ a_1, a_2 \in Z_\beta;\ h_{(p,q,\alpha,\beta,\gamma)}(a_1, a_2) = \alpha^{a_1}\gamma^{a_2} \pmod n.$$

Discrete Logarithm (DL) assumption for a general finite group [16]

Given $I = (n, \alpha, \beta)$, where n is a composite number, α is an element of Z_n^* and $\beta \in Z_n^*$, where

$$\alpha^a \equiv \beta \pmod n$$

it is hard to find an integer $a = log_\alpha \beta$.

Theorem 3.1. *Under the above DL assumption [16], the above construction is a family of bundling homomorphisms.*

Proof. To show that the above definition is a bundling homomorphism, we have to show that (definition 4.1 [17]),

(1) For any $\mu \in Z_n^*$ where $\mu = \alpha^{a_1}\gamma^{a_2} \pmod n$, $(a_1, a_2) \in Z_\beta \times Z_\beta$, there are β preimages in Z_β.
(2) For a given $\mu \in Z_n^*$ where $\mu = \alpha^{a_1}\gamma^{a_2} \pmod n$, $(a_1, a_2) \in Z_\beta \times Z_\beta$, it is difficult to find a pair $(\tilde{a_1}, \tilde{a_2})$ such that $\alpha^{\tilde{a_1}}\gamma^{\tilde{a_2}} = \mu \pmod n$.
(3) It is hard to find two pairs (a_1, a_2), $(\tilde{a_1}, \tilde{a_2}) \in Z_\beta \times Z_\beta$ that map to the same value.

To prove property 1, we note that knowing $\mu = \alpha^k \pmod n = \alpha^{a_1}\gamma^{a_2} \pmod n$ for $\gamma = \alpha^a \pmod n$ and $ord_n(\alpha) = \beta$, there exists exactly β different values of $(\tilde{a_1}, \tilde{a_2})$ in Z_β that satisfy $k = \tilde{a_1} + a\tilde{a_2} \pmod \beta$. Hence there are β preimages for μ in Z_β.

Now given $\mu = \alpha^{a_1 + aa_2} \pmod n$, finding $a_1 + aa_2$ is equivalent to solving an instance of DL problem [16], which is hard (property 2).

Property 3 means that it is difficult to find (a_1, a_2) and $(\tilde{a_1}, \tilde{a_2})$ such that $\alpha^{a_1}\gamma^{a_2} = \alpha^{\tilde{a_1}}\gamma^{\tilde{a_2}} \pmod n$. Suppose that there is a probabilistic polynomial-time algorithm $\tilde{A}$ that could compute such a collision. Then, we construct an algorithm $\tilde{D}$ that on input $(n, \alpha, \beta, \gamma)$, where $\gamma = \alpha^a \pmod n$, outputs the secret value a as follows:

First, $\tilde{D}$ runs $\tilde{A}$, and if $\tilde{A}$ outputs a collision, i.e. (s_1, s_2) and $(\tilde{s_1}, \tilde{s_2})$, such that $\alpha^{s_1}\gamma^{s_2} = \alpha^{\tilde{s_1}}\gamma^{\tilde{s_2}} \pmod n$, then $\tilde{D}$ computes:

$$\begin{aligned}
&\alpha^{s_1}\gamma^{s_2} = \alpha^{s'_1}\gamma^{s'_2} \pmod n\\
&\alpha^{s_1 + as_2} = \alpha^{s'_1 + as'_2} \pmod n\\
&\alpha^{s_1 - s'_1} = \alpha^{a(s'_2 - s_2)} \pmod n\\
&s_1 - s'_1 = a(s'_2 - s_2) \pmod \beta\\
&a = (s_1 - s'_1)(s'_2 - s_2)^{-1} \pmod \beta.
\end{aligned}$$

$\tilde{D}$ is successful with the same probability as $\tilde{A}$ and almost equally efficient. Hence, it contradicts with the DL assumption [16]. ■

Theorem 3.2. *The FSS scheme described above is secure for the signer.*

According to the Theorem 4.2 in [17], we must find the size of the set T:

$$T := \{(c_1, c_2) \in Z_\beta \times Z_\beta | \alpha^{c_1}\gamma^{c_2} = 1 \pmod n \wedge (m'(c_1 + ac_2) = 0)\}$$

for all values of m' between 1 and $\beta - 1$, given that the prekey is good. Since $(0, 0)$ is the only element of this set, then the size of the set T is 1.

Together with theorem 4.2 [17], this implies that it suffices to choose $\tau = \sigma$ in the proposed scheme. ■

3.2. Efficiency comparison

In this section we compare efficiency of the proposed scheme with the best known FSS schemes. Efficiency of an FSS scheme has been measured in terms of three length parameters: the lengths of the secret key, the public key and the signature, and the amount of computation required in each case. To compare two FSSs we fix the level of security provided by the two schemes and find the size of the three length parameters, and the number of operations (for example multiplication) required for signing and testing.

Table 1 gives the results of comparison of four FSS schemes when the security levels of the receiver and the sender are given by k and σ, respectively. In this comparison, the first two schemes (first and second column of the table) are chosen because they have provable security. The first scheme, proposed by van Heijst and Pedersen [29], is the most efficient and provably secure scheme, which is based on discrete logarithm assumption. We refer this scheme as DL scheme in this paper.

Table 1
Comparison of efficiency parameters.

	DL [29]	*Fact* [30,17]	*RSA based* [28]	*Our FSS*
PK (mult)	4K	2K	4K	4K
Sign (mult)	2	K	2	2
Test (mult)	3K	$2K + \sigma$	3K	3K
Length of SK (bits)	4K	$4K + 2\sigma$	4K	4K
Length of PK (bits)	$2\hat{K}$	2K	2K	$2\hat{K}$
Length of a signature (bits)	2K	$2K + \sigma$	4K	2K
Underlying hard problem	DL	Factorization	Factorization	Factorization

The second scheme is a factorization based FSS proposed in [30,17]. The third scheme is the RSA based FSS scheme [28]. This scheme is included for completeness and to represent an FSS scheme based on factorization assumption. Column four corresponds to our proposed scheme.

We note that there are several other schemes in the literature such as [21,22,26,27,24]. Nonetheless, we do not include those schemes and rather select the above three schemes for the following reasons. The schemes in [21,22] are generic construction of FSS schemes from A-codes, and therefore the resulting constructions are rather inefficient. Note that these works provide the second fundamental result on FSS schemes in addition to the seminal paper of [17]. The schemes in [26, 27,24] are all based on factorization assumption. Susilo et al. [27] proposed an efficient factorization assumption FSS scheme which was later on shown to be insecure in [24] if the parameters are not chosen carefully. Samoa [24] offered a solution to fix the scheme in [27] but the scheme will be rather inefficient. Furthermore, Samoa [24] also presented a secure factorization-based FSS scheme that relies on a composite number $n = p^2q$, and therefore the result is rather inefficient. Independently, Susilo and Mu [26] proposed a factorization-based FSS scheme that uses the Hensel-RSA technique. Nonetheless, this scheme is also inefficient since the resulting signature size is rather large.

We use the same value of σ and k for all the systems and determine the size of the three length parameters. The hard underlying problem in all three schemes are Discrete Logarithm (DL) problem, Subgroup DL [15] and/or Factorization problem. This means the same level of receiver's security (given by the value of parameter k) translates into different size primes and moduli. In particular, the security level of a 151 bits subgroup discrete logarithm with basic primes of at least 1881 bits, is the same as factorization of a 1881 bits RSA modulus [15].

To find the required size of primes in DL scheme, assuming security parameters (k, σ) are given, first $K = \max(k, \sigma)$ is found and then the prime q is chosen such that $|q|_2 \geq K$. The bundling degree in this scheme is q and the value of p is chosen such that $q|p - 1$ and $(p - 1)/q$ be upper-bounded by a polynomial in K (page 237 and 238 [19]). The size of $|p|_2$ must be chosen according to standard discrete logarithm problem, which for adequate security must be at least 1881 bits [15]. However, the size of $|q|_2$ can be chosen as low as 151 bits [15]. Since $|p|_2$ and $|q|_2$ are to some extent independent, we use $\hat{K}$ to denote $|p|_2$.

In the factorization scheme of [17], the security level of the sender, σ satisfies $\tau = \rho + \sigma$ where τ is the bundling degree and 2^ρ is the size of the message space. Security parameter of the receiver, k, is determined by the difficulty of factoring the modulus n. Now for a given pair of security parameters, (k, σ), the size of modulus N_k is determined by k but determining τ requires knowledge of the size of the message space. Assume $\rho = |p|_2 \approx |q|_2 = N_k/2$. This means that $\tau = \sigma + N_k/2$. Now the efficiency parameters of the system can be given as shown in the table. In particular the size of secret and public keys are $2(\tau + N_k)$ and $2N_k$ respectively.

In RSA-based FSS scheme [28], $\tau = |\phi(n)|_2$, and security of the receiver is determined by the difficulty of factoring n. This means that $\tau \approx |n|_2$. To design a system with security parameters (k, σ), first N_k, the modulus size that provides security level k for the receiver is determined and then $K = \max(\sigma, |N_k|_2)$. The modulus n is chosen such that $|n|_2 = K$. With this choice, the system provides adequate security for the sender and the receiver.

In our proposed scheme bundling degree, and hence security level of the sender is $\sigma = \tau = |\beta|_2$. The security of the receiver is determined by the difficulty of factorization of n and discrete logarithm in a subgroup of size β in Z_n^*. Assume $|p|_2 \approx |q|_2 \approx \frac{|n|_2}{2}$ and $n \approx c \times |\beta|_2$. Then we first find N_k which is the modulus size for which factorization has difficulty k. Next, we find F_{k,N_k} which is the minimum size of a multiplicative subgroup of Z_n^* for which subgroup discrete logarithm has hardness k. Finally, we choose $K = max(F_{k,N_k}, \sigma)$ and set $|\beta|_2 = K$. With these choices, the sender and receiver's level of security is at least σ and k, respectively. We use $\hat{K}$ to represent $|n|_2$.

The proposed scheme is more efficient than the factorization scheme of [28] and [17] and is as efficient as the DL scheme. In DL scheme, to achieve the adequate security, K must be chosen to be at least 151 bits, and $\hat{K}$ must be at least 1881 bits [15]. These are also the values required by our scheme.

We note that if we move the DL scheme to the pairing group, then we will not achieve a better result. As shown in the above table, the DL scheme will require the size of the signature to be 2K. For completeness, the bilinear pairing is reviewed as follows. The bilinear pairing e that will be used is the *admissible* bilinear pairing, which is defined over two groups of the same prime-order q denoted by $\mathbb{G}_1$ and $\mathbb{G}_2$. Suppose that $\mathbb{G}_1$ is generated by g. Then, $e : \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2$ has the following properties: (1) Bilinear: $e(g^a, g^b) = e(g, g)^{ab}$, for all $a, b \in \mathbb{Z}_q$ and (2) Non-degenerate: $e(g, g) \neq 1$. In the bilinear pairing

version of the above DL scheme[2] [29], the bundling homomorphism will map $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, which implies that the size of the signature will be $2K = 2|\mathbb{G}_2| \approx 2 \times 160$ bits $= 320$ bits (cf. 302 bits in our scheme).

4. Conclusions

We constructed a short FSS scheme based on factorization and discrete logarithm which is provably secure. Interestingly, although our scheme uses factorization assumption, we can achieve a shorter signature compared to the existing schemes based on the same assumption. We note that in the existing literature, all FSS schemes based on factorization will produce a very long signature size compared to the FSS schemes that are based on discrete logarithm assumption. Furthermore, our scheme does not incorporate any pairing operations (in contrast to any other signature schemes with special properties which use these operations).

References

[1] N. Barić, B. Pfitzmann, Collision-free accumulators and fail-stop signature schemes without trees, in: Advances in Cryptology — Eurocrypt'97, in: Lecture Notes in Computer Science, vol. 1233, 1997, pp. 480–494.
[2] D. Boneh, X. Boyen, Short signatures without random oracles, in: Advances in Cryptology — Eurocrypt 2004, in: Lecture Notes in Computer Science, vol. 3027, 2004, pp. 56–73.
[3] D. Boneh, X. Boyen, H. Sacham, Short group signature, in: Advances in Cryptology — Crypto 2004, in: Lecture Notes in Computer Science, vol. 3152, 2004, pp. 41–55.
[4] D. Boneh, M. Franklin, Identity-based encryption from the Weil pairing, in: Lecture Notes in Computer Science, vol. 2139, 2001, p. 213+. URL citeseer.nj.nec.com/article/boneh01identitybased.html.
[5] D. Boneh, C. Gentry, B. Lynn, H. Shacham, Aggregate and verifiable encrypted signatures from bilinear maps, in: Proceedings of Eurocrypt 2003, in: Lecture Notes in Computer Science, vol. 2656, 2003, pp. 416–432.
[6] D. Boneh, B. Lynn, H. Shacham, Short signatures from the weil pairing, in: Advanced in Cryptology — Asiacrypt 2001, in: Lecture Notes in Computer Science, vol. 2248, Springer Verlag, 2001, pp. 514–532.
[7] E. Brickell, K. McCurley, An interactive identification scheme based on discrete logarithms and factoring, in: Advances in Cryptology — Eurocrypt'90, in: Lecture Notes in Computer Science, vol. 437, 1991, pp. 63–71.
[8] E. Brickell, K. McCurley, An interactive identification scheme based on discrete logarithms and factoring, Journal of Cryptology 5 (1) (1992) 29–39.
[9] H.K.-C. Chang, E.-H. Lu, P.-C. Su, Fail-stop blind signature scheme design based on pairings, Applied Mathematics and Computation 169 (2) (2004) 1324–1331.
[10] D. Chaum, E. van Heijst, B. Pfitzmann, Cryptographically strong undeniable signatures, unconditionally secure for the signer, Interner Bericht, Fakultät für Informatik 1/91.
[11] X. Chen, F. Zhang, W. Susilo, Y. Mu, Efficient generic on-line/off-line signatures without key exposure, in: The 5th International Conference on Applied Cryptography and Network Security, ACNS'07, in: Lecture Notes in Computer Science, vol. 4521, 2007, pp. 18–30.
[12] W. Diffie, M. Hellman, New directions in cryptography, IEEE IT 22 (1976) 644–654.
[13] X. Huang, Y. Mu, W. Susilo, W. Wu, Provably secure pairing-based convertible undeniable signature with short signature length, in: International Conference on Pairing-based Cryptography (Pairing 2007), in: Lecture Notes in Computer Science, vol. 4575, Springer-Verlag, 2007, pp. 367–391.
[14] X. Huang, W. Susilo, Y. Mu, F. Zhang, Short designated verifier signature scheme and its identity-based variant, International Journal of Network Security (IJNS) 6 (1) (2003) 82–93.
[15] A. Lenstra, E. Verheul, Selecting cryptographic key sizes, online: http://www.cryptosavvy.com/. Extended abstract appeared in Commercial Applications, Price Waterhouse Coopers, CCE Quarterly Journals 3 (1999) 3–9.
[16] T. Okamoto, K. Sakurai, H. Shizuya, How intractable is the discrete logarithm for a general finite group? in: Advances in Cryptology Eurocrypt 1992, in: Lecture Notes in Computer Science, vol. 658, Springer-Verlag, Berlin, 1992, pp. 420–428.
[17] T.P. Pedersen, B. Pfitzmann, Fail-stop signatures, SIAM Journal on Computing 26/2 (1997) 291–330.
[18] B. Pfitzmann, Fail-stop signatures without trees, Hildesheimer Informatik-Berichte, Institut für Informatik 16/94.
[19] B. Pfitzmann, Digital Signature Schemes–General Framework and Fail-Stop Signatures, in: Lecture Notes in Computer Science, vol. 1100, Springer-Verlag, 1996.
[20] B. Pfitzmann, M. Waidner, Formal aspects of fail-stop signatures, Interner Bericht, Fakultät für Informatik 22/90.
[21] R. Safavi-Naini, W. Susilo, A general construction for fail-stop signature using authentication codes, in: Proceedings of Workshop on Cryptography and Combinatorial Number Theory, CCNT'99, Birkhäuser, 2001, pp. 343–356.
[22] R. Safavi-Naini, W. Susilo, General construction of fail-stop signature schemes based on authentication codes, in: K.-Y. Lam, I.E. Shparlinski, H. Wang, C. Xing (Eds.), Progress in Computer Science and Applied Logic, vol. 20, Birkhauser, 2001, pp. 343–356.
[23] R. Safavi-Naini, W. Susilo, H. Wang, An efficient construction for fail-stop signatures for long messages, Journal of Information Science and Engineering (JISE) — Special Issue on Cryptology and Information Security 17 (2001) 879–898.
[24] K. Schmidt-Samoa, Factorization-based fail-stop signatures revisited, in: Information and Communications Security, ICICS 2004, in: Lecture Notes in Computer Science, vol. 3269, Springer-Verlag, Berlin, 2004, pp. 118–131.
[25] V. Shoup, A Computational Introduction to Number Theory and Algebra, Cambridge University Press, 2005, pp. 263–264.
[26] W. Susilo, Y. Mu, Provably secure fail-stop signature schemes based on rsa, International Journal of Wireless and Mobile Computing (IJWMC, Inderscience Publishers) 1 (1) (2005) 53–60.
[27] W. Susilo, R. Safavi-Naini, M. Gysin, J. Seberry, A new and efficient fail-stop signature schemes, The Computer Journal 43 (5) (2000) 430–437.
[28] W. Susilo, R. Safavi-Naini, J. Pieprzyk, RSA-based fail-stop signature schemes, in: International Workshop on Security, IWSEC'99, IEEE Computer Society Press, 1999, pp. 161–166.
[29] E. van Heijst, T. Pedersen, How to make efficient fail-stop signatures, in: Advances in Cryptology — Eurocrypt'92, 1992, pp. 366–377.
[30] E. van Heijst, T. Pedersen, B. Pfitzmann, New constructions of fail-stop signatures and lower bounds, in: Advances in Cryptology — Crypto'92, in: Lecture Notes in Computer Science, vol. 740, 1993, pp. 15–30.
[31] M. Waidner, B. Pfitzmann, The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability, in: Advances in Cryptology — Eurocrypt'89, in: Lecture Notes in Computer Science, vol. 434.
[32] H. Xiaoming, H. Shangteng, Comment fail-stop blind signature scheme design based on pairings, Wuhan University Journal of Natural Sciences (2006) 1545–1548.
[33] F. Zhang, R. Safavi-Naini, W. Susilo, An efficient signature scheme from bilinear pairings and its application, in: The 7th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2004, in: Lecture Notes in Computer Science, vol. 2947, 2004, pp. 277–290.
[34] F. Zhang, W. Susilo, Y. Mu, Identity-based partial message recovery signatures (or how to shorten ID-based signatures), in: Financial Cryptography and Data Security, FC'05, in: LNCS, vol. 3570, 2005, pp. 45–56.

[2] Note that this can be achieved trivially by following the original van Heijst and Pedersen's scheme but using the bilinear group instead of the finite fields.